Journal of Computational and Applied Mathematics 21 (1988) 333–341
North-Holland


# The partial total least squares algorithm

Sabine VAN HUFFEL and Joos VANDEWALLE
*ESAT-Laboratory, Department of Electrical Engineering, K.U. Leuven, B-3030 Heverlee, Belgium*

Received 20 August 1987

*Abstract:* In this paper, an improved algorithm PTLS for solving total least squares (TLS) problems $AX \approx B$ is presented. As only a basis of the right singular subspace associated with the smallest singular values of the data $[A; B]$ is needed, the computational cost can be reduced considerably by using the partial SVD algorithm. This algorithm computes in an efficient way a basis for the left and/or right singular subspace of a matrix associated with its smallest singular values.

An analysis of the operation counts, as well as computational results, show the relative efficiency of PTLS with respect to the classical TLS algorithm. Typically, PTLS reduces the computation time with a factor 2.

*Keywords:* Total least squares, overdetermined sets of equations, numerical linear algebra, singular subspace, singular value decomposition.

## 1. Introduction

Total least squares (TLS) is one method of solving overdetermined sets of linear equations $AX \approx B$ that is appropriate when there are errors in both the observation matrix $B$ and the data matrix $A$.

The TLS problem can be formulated as follows ($R$ denotes the range):

**TLS definition.** Given an overdetermined set of $m$ linear equations in $n \times d$ unknowns $X$

$$AX \approx B, \qquad A \in \mathbb{R}^{m \times n}, \quad B \in \mathbb{R}^{m \times d}, \quad X \in \mathbb{R}^{n \times d}, \tag{1}$$

then, a TLS solution is any solution $X$ of the set

$$\hat{A}X = \hat{B} \tag{2}$$

where $\hat{A}$ and $\hat{B}$ are determined such that

$$R(\hat{B}) \subset R(\hat{A}), \tag{3}$$

$$\| [\Delta\hat{A}; \Delta\hat{B}] \|_F = \| [A; B] - [\hat{A}; \hat{B}] \|_F \quad \text{is minimal.} \tag{4}$$

The problem of finding $[\Delta\hat{A}; \Delta\hat{B}]$ such that (3)–(4) are satisfied, is referred to as the TLS problem.

Whenever the TLS solution is not unique, TLS singles out the minimum norm solution. Golub and Van Loan (1980) introduced this method into the field of numerical analysis and developed

an algorithm based on the singular value decomposition (SVD). However in some TLS problems, called nongeneric, their algorithm fails to compute a finite TLS solution. We generalized their TLS computations in order to solve these nongeneric TLS problems [12; 11, Section 1.6] and summarized the different TLS computations into one algorithm, called the classical TLS algorithm, which includes the proposed generalization [10; 11, Section 1.8.1].

Although TLS can improve the solution accuracy, its computational cost can be too high in applications where the computational speed is an important factor, e.g. in real-time estimation. Therefore, we now improve the efficiency of the TLS computations as presented in the classical TLS algorithm [6; 11, Section 1.8.1]. Since the TLS solution of (1) is deduced from only one right singular vector or in general, a basis of the right singular subspace associated with the smallest singular values of the data $[A; B]$, a considerable saving in computation time is possible by only calculating those desired base vectors. This can be done by using the partial SVD algorithm [13]. This algorithm calculates in an efficient and reliable way a basis for the left and/or right singular subspace of a matrix associated with its smallest singular values. The dimension of the desired subspace may be fixed or depend on a given parameter.

There are three reasons for its high efficiency with respect to the classical SVD algorithm [5]. First, the Householder transformations of the bidiagonalization are only applied to the base vectors of the desired singular subspace. Second, the bidiagonal is only partially diagonalized and third, an appropriate choice is made between QR and QL iteration steps. See [13; 11, Section 9] for a detailed analysis of this method.

Based on this algorithm PSVD, the TLS computations are speeded up resulting in an improved 'partial TLS' algorithm PTLS which is outlined in Section 2. Section 3 compares PTLS with the classical TLS algorithm. An analysis of the operation counts, as well as computational results, show the relative efficiency of PTLS with respect to the classical TLS algorithm. Finally, Section 4 presents the conclusions.

## 2. Outline of the partial TLS algorithm PTLS

The computational efficiency of the classical TLS algorithm, presented in [6; 11, Section 1.8.1] can be improved by using the 'partial SVD' instead of the classical SVD algorithm. Indeed, as shown in [11, Section 1.8.1] the TLS solution of (1) can be obtained from any orthogonal basis of the right singular subspace associated with the smallest $n + d - r$ singular values of the $m$ by $n + d$ matrix $[A; B]$ of rank $r$. This improvement is incorporated in the following 'partial TLS' algorithm PTLS.

**Algorithm 2.1.** Partial TLS (PTLS) computation of the TLS solution $\hat{X}$ of $AX \approx B$;

*Given:*
- an $m$ by $n$ data matrix $A$ and an $m$ by $d$ observation matrix $B$.
- an upper bound $\theta$ on the smallest singular values $\sigma_i$ of $[A; B]$ ordered in decreasing order of magnitude (such that $\sigma_r[A; B] > \theta \geqslant \sigma_{r+1}[A; B]$) or the desired rank $r$ of $[A; B]$ with an initial estimate of $\theta$ (the dimension of the desired singular subspace is then $n + d - r$);
- and a stop criterion $\delta$ defining the minimum subinterval length in the bisection method (Step 4 in Algorithm 2.2).

*Step 1.* Bidiagonalization phase. Transform $[A; B]$ into bidiagonal form $J$ by performing Step 1 of the PSVD algorithm [13].

*Step 2.* Partial diagonalization phase. Diagonalize $J$ partially by performing Step 2 of the PSVD algorithm [13].

*Step 3.* Back transformation phase. Apply the Householder transformations of the bidiagonalization (Step 1) to each base vector $v_i$ of the desired right singular subspace by performing Step 3 of the PSVD algorithm [13].

*Step 4.* Compute a Householder matrix $Q$ such that

$$[v_{r+1}, \ldots, v_{n+d}]Q = \begin{bmatrix} \square & Z \\ 0 & \Gamma \end{bmatrix} \begin{matrix} n \\ d \end{matrix}$$
$$\qquad n-r \quad d$$

with $v_{r+1}, \ldots, v_{n+d}$ the base vectors of the desired right singular subspace and $\Gamma$ a $d$ by $d$ upper triangular matrix.

*Step 5.* If $\Gamma$ is singular then begin
lower the rank $r$ with one
go back to Step 2
end
else solve $\hat{X}\Gamma = -Z$ by forward elimination

*End.*

**Remarks.** (i) This algorithm has been programmed in Fortran, fully documented and tested. A program listing is given in [10].

(ii) In case the rank must be lowered in Step 5, a new upper bound $\theta$ on the singular values associated with the enlarged desired singular subspace, must be computed. Hereto, the PSVD algorithm uses a bisection method [7, Section 8.5] at the end of the bidiagonalization phase to compute an appropriate bound $\theta$ such that the matrix $[A; B]$ has exactly $n + d - r$ singular values smaller than or equal to $\theta$ (see Algorithm 2.2). Once the bound has been computed, the additional base vectors can be computed. The additional computations can be kept to a minimum. Indeed, we only have to further diagonalize the partially reduced bidiagonal by zeroing out some more superdiagonal elements (Step 2) and back transforming these newly computed base vectors (Step 3). The base vectors which were already computed, remain unaffected.

(iii) If the rank $r$ is lowered in Step 5 but $\sigma_r$ is of multiplicity $> 1$, the rank $r$ must be further reduced with the multiplicity of $\sigma_r$. This is done in the beginning of Step 2 by $\theta$-estimation Algorithm 2.2 given below (to be used as auxiliary routine in the PSVD algorithm). Hereto, a stop criterion $\delta$ must be entered defining the minimum subinterval length for the bisection method. The multiplicity of $\sigma_r$ is then determined by the number of singular values lying within an interval of length $\delta$ around $\sigma_r$.

**Algorithm 2.2.** Estimation of the bound $\theta$.

*Given:*
- an $n$ by $n$ bidiagonal $J$, defined by its diagonal elements $q_1,, \ldots, q_n$ and its superdiagonal elements $e_2, \ldots, e_n$.
- the dimension $d$ of the desired singular subspace of $J$.

- an initial estimate of $\theta$ ($\theta \leqslant \delta$ if no initial estimate is available).
- a stop criterion $\delta$, defining the minimum subinterval length in the bisection method.

*Step 1.* Initialization of $\theta$.
If $\theta \leqslant \delta$ then if $d = 1$ then $\theta \leftarrow \min_{i=1,n} |q_i|$
else $\theta \leftarrow |q_{n-d+1}|$
*Step 2.* check quality initial estimate of $\theta$.
num $\leftarrow \#\{\sigma_i(J) \mid \sigma_i \leqslant \theta\}$
if num $= d$ return
*Step 3.* Initialization starting values for bisection method.
If num $< d$ then $y \leftarrow \theta$; $z \leftarrow \max_{i=2,n}(|q_i| + |e_i|, |q_{i-1}| + |e_i|)$
else $z \leftarrow \theta$; $y \leftarrow \delta$
*Step 4.* Bisection method.
Do while num $\neq d$
$\theta \leftarrow \frac{1}{2}(y + z)$
num $\leftarrow \#\{\sigma_i(J) \mid \sigma_i \leqslant \theta\}$
if num $< d$ then $y \leftarrow \theta$
else $z \leftarrow \theta$
if $z - y < \delta$ then begin
$\theta \leftarrow z$
$d \leftarrow \#\{\sigma_i(J) \mid \sigma_i \leqslant \theta\}$
return
end
end
*End.*

**Remarks.** (i) In Step 1, using the properties of triangular matrices [8] we know that min $|q_i|$ is always an upper bound for the minimal singular value of $J$ and approaches it very closely, when the gap is large, i.e. $\sigma_{n-1}(J) \gg \sigma_n(J)$. The initial estimate of $\theta$ in case $d > 1$ was experimentally found to be suitable if the gap between $\sigma_{n-d}(J)$ and $\sigma_{n-d+1}(J)$ is large (this is the case for most TLS problems).

(ii) In Step 2 and 4, the number of singular values which are $\leqslant \theta$ is the sum of the number of singular values $\leqslant \theta$ over all unreduced subbidiagonals $J_j$ of $J$. The computation of the number of singular values $\leqslant \theta$ of each unreduced subbidiagonal $J_j$ is based on applying Sylvester's Law of Inertia, or equivalently, Sturm sequences [9, p. 52; 7, Section 8.5] to the unreduced symmetric tridiagonal matrices associated with each unreduced subbidiagonal $J_j$. Hereto, the tridiagonal matrices $T_j = J_j^T J_j$ whose eigenvalues are the squared singular values of $J_j$, can be used [13; 4, p. 213]. A numerically more accurate and equally efficient method is to identify the singular value problem for the bidiagonal $J_j$ with the eigenvalue problem for the following symmetric matrix:

$$J_j' = \begin{bmatrix} 0 & J_j^T \\ J_j & 0 \end{bmatrix}. \tag{5}$$

The eigenvalues of $J_j'$ are the singular values of $J_j$ and their negatives [7, p. 286]. Suppose that the bidiagonal $J_j$ is $k$ by $k$ with diagonal entries $q_1, \ldots, q_k$ and superdiagonal entries $e_2, \ldots, e_k$.

Then by permuting the rows and columns of $J_j'$ to appear in the new order 1, $k+1$, 2, $k+2, \ldots, k$, $2k$ we see that $J_j'$ is orthogonally similar to the tridiagonal matrix $J_j''$ with zeros on its diagonal and offdiagonals $q_1$, $e_2$, $q_2$, $e_3, \ldots, e_k$, $q_k$ [2, p.5; 4, p.213]. Thus the singular values of $J_j$ are the absolute values of the eigenvalues of the matrix $J_j''$ onto which Sylvester's Law of Inertia can be applied. Then, the number of singular values of $J_j$ less than or equal to $\theta$ is precisely the number of nonpositive $d_i$ minus $k$, where the $d_i$ are given by the following recurrence [9, p.47]:

$$d_1 = -\theta, \qquad d_i = -\theta - b_{i-1}^2/d_{i-1},$$

with $b_1 = q_1$, $b_2 = e_2, \ldots, b_{2k-2} = e_k$, $b_{2k-1} = q_k$ the offdiagonals of $J_j''$. If $J_j''$ has nonzero offdiagonals it must have simple eigenvalues [9, p. 124] and $d_{2k} = 0$ if and only if $\theta$ is an eigenvalue of $J_j''$.

This method is accurate as proven by Kahan and Demmel in [2,4]. If the computed number of nonpositive $d_i$ is $k+s$, there must be at least s singular values of $J_j$ less than or equal to $\theta/(1-(3k-1.5)\epsilon)$ and no more than $s$ singular values less than or equal to $\theta(1-(6k-2)\epsilon)/(1-(3k-1.5)\epsilon)$ where $\epsilon$ is the machine precision.

(iii) In Step 3, the initial upper bound for the bisection method is derived from the Gerschgorin circle theorem [7, Theorem 7.2-1] applied to all $J_j'$ in (5).

(iv) In Step 4, the bisection method produces a sequence of subintervals $[y, z]$ that are repeatedly halved in length but such that

$$\#\{\sigma_i(J) \mid \sigma_i \leqslant y\} < d < \#\{\sigma_i(J) \mid \sigma_i \leqslant z\}.$$

We have to find a bound $\theta$ such that $\sigma_{n-d}(J) > \theta \geqslant \sigma_{n-d+1}(J)$. Hence, if at a certain moment $z - y < \delta$ then at least two singular values of $J$ lie in the interval $[y, z]$ within a distance $< \delta$ from each other. If $\delta$ is defined by the standard deviation of the noise on the data $[A; B]$, this means that $\sigma_{n-d+1}$ may be considered as a singular value of multiplicity $> 1$. Hence, it makes no sense to continue the bisection further, trying to separate the singular values $\sigma_{n-d}$ and $\sigma_{n-d+1}$ of $J$. Instead, we consider the singular values of $J$ in the interval $[y, z]$ as coinciding with $\sigma_{n-d+1}$ and determine its multiplicity by the number of singular values of $J$ in the interval $[y, z]$. We therefore better enlarge the dimension of the desired singular subspace such that it contains all base vectors of the $\rho$-dimensional singular subspace associated with $\sigma_{n+d-1}(J)$ of multiplicity $\rho$. Observe however that this method only allows to compute the multiplicity of a singular value rather arbitrarily and does not allow to check the multiplicity of singular values as efficiently and reliably as the classical TLS algorithm [11, Section 1.8.1].

## 3. Comparison of the classical TLS versus the partial TLS algorithm

Since the TLS computations are entirely dominated by the SVD, it is evident that a comparison between classical TLS and PTLS reduces mainly to a comparison of the classical SVD with PSVD, for computing a right singular subspace (i.e. case a discussed in [13; 11, Section 9]. However, some additional comments are in order.

### *3.1. Storage requirements*

Concerning the storage requirements, the PTLS Algorithm 2.1 needs more memory space, namely $\frac{1}{2}(n+d)(n+d-1)$ (resp., $m(n+d-\frac{1}{2}(m+1))$) extra storage locations to solve (1) in

case $m \geq n + d$ (resp., $m < n + d$). Indeed, the Householder transformations performed onto the rows of $[A; B]$ during its bidiagonalization (see Step 1) must be stored until the end of the computations (Step 5). The classical TLS algorithm accumulates the products of the Householder transformations immediately into the right singular base matrix $V$ after the bidiagonalization. Hence, they need not be stored and matrix $[A; B]$ can be used to store $V$.

### 3.2. Operation counts

W.r.t. the operation counts, the following comments are worth mentioning. Once the desired base vectors of $[A; B]$ are computed by classical SVD or PSVD, the TLS solution of (1) must be calculated. Hereto we first reduce those base vectors with Householder transformations to the form (5) (Step 4 of Algorithm 2.1) and then solve

$$\hat{X}\Gamma = -Z \tag{6}$$

by forward elimination (Step 5 of Algorithm 2.1). If the rank of $[A; B]$ equals $n$, $d$ base vectors have been calculated. In this case, the computation of $\Gamma$ and the TLS solution from (6) (Step 4 and 5 of Algorithm 2.1) requires

$$\tfrac{1}{6}\left[(9n + 4d + 12)d^2 + (15n + 26)d\right] - 3n - 7 \tag{7}$$

multiplications (and divisions). Observe that (7) reduces to $n$ if $d = 1$ in (1). This is indeed the number of multiplications required to scale the $(n + 1)$th right singular vector of $[A; B]$ and obtain the TLS solution.

The number of operations required for solving TLS problems (1) with $A$ of full rank, are then immediately obtained. Hereto, we add (7) to the operation counts of the classical SVD and PSVD (see [13, Table 1, case a]) and replace $n$ by the number of columns $n + d$ in $[A; B]$. We then obtain the expected number of multiplications (and divisions) for the classical TLS and PTLS algorithm. The counts of the classical SVD are based on the SVD routine DSVDC of the LINPACK library [3, Section 11]. If $m \geq \frac{5}{3}(n + d)$, matrix $[A; B]$ is first reduced to triangular form $R$ in order to improve the efficiency (according to [1]). This is done with the LINPACK routine DQRDC [3, Section 9]. Extra calculations which deal with under- and overflow, are not taken into account.

We assume that $m \geq n$ and only consider the third and second order terms in $m$ and $n$. So the results are correct for moderate $m$ and $n$. The average number of QR iteration steps for each singular value during the diagonalization in the classical SVD algorithm is denoted by $s$. The average number of QR/QL iteration steps needed for convergence to one base vector in PSVD is denoted by $z$ and $d$ is the dimension of the desired singular subspace in PSVD.

Classical TLS:

$$\begin{aligned} m < \tfrac{5}{3}(n + d)\colon\ & 2m(n + d)^2 + 2s(n + d)^3 + (\tfrac{3}{2} + 8s)(n + d)^2 \\ & + \tfrac{1}{6}(9n + 4d + 12)d^2 + \tfrac{5}{2}nd, \\ m \geq \tfrac{5}{3}(n + d)\colon\ & m(n + d)^2 + (\tfrac{5}{3} + 2s)(n + d)^3 + (2 + 8s)(n + d)^2 \\ & + m(n + d) + \tfrac{1}{6}(9n + 4d + 12)d^2 + \tfrac{5}{2}nd; \end{aligned}$$

Partial TLS:

$$m < \tfrac{5}{3}(n+d)\colon\ 2m(n+d)^2 - 2(n+d)^3/3 + \left[(4z+1)d+2\right](n+d)^2 + \tfrac{1}{6}(9n+4d+12)d^2 + \tfrac{5}{2}nd,$$

$$m \geqslant \tfrac{5}{3}(n+d)\colon\ m(n+d)^2 + (n+d)^3 + \left[(4z+1)d+\tfrac{5}{2}\right](n+d)^2 + m(n+d) + \tfrac{1}{6}(9n+4d+12)d^2 + \tfrac{5}{2}nd;$$

### 3.3. Computational results

Finally, we give some computational results. Observe that the comparison between classical TLS and PTLS largely agrees with the computational results of the classical SVD and PSVD given in [13]. Indeed, when $d \ll n$, the additional computations (7) to obtain the TLS solution, are negligible w.r.t. the computations of the desired base vectors by classical SVD or PSVD. As classical TLS algorithm we use the routine given in [10]. This routine uses the SVD routine of the LINPACK library [3, Section 11]. The matrices $[A;\ B]$ were randomly generated with specified singular value spectrum. All tests were run on the IBM 3030 of the Central Computer Center at the Katholieke Universiteit Leuven. In the examples given below, $\epsilon$ denotes the numerical accuracy. Double precision was used throughout the calculations. The CPU times are expressed in milliseconds. $V$ denotes the matrix containing the $n+d$ right singular vectors of $[A;\ B]$.

**Example 1.** Consider the equations (1) with $A$ a 30 by 16 matrix and one observation vector $b$ ($d = 1$). The singular values of $[A;\ B]$ whose rank is 16, are 1000, 800, 400, 200, 100, 80, 40, 20, 10, 8, 4, 2, 1, 0.8, 0.4, 0.2 and 0.01. Hence, only one solution vector must be computed. $\epsilon = 10^{-13}$. 3 QR iterations were required for convergence. Table 1 shows the improvement in CPU time for the steps in the algorithm.

**Example 2.** Consider the set of equations (1) with $A$ a 15 by 10 matrix and 3 observation vectors ($d = 3$). The singular values of $[A;\ B]$ whose rank is 10, are 100, 50, 28, 26, 24, 22, 20, 18, 15, 12,

Table 1
Comparison of CPU times (in msec.) for solving Example 1

| | PTLS | Classical TLS |
|---|---|---|
| triangularization of $[A;\ b]$ | 21 | 21 |
| bidiagonalization of $R$ | 28 | 28 |
| initialization $V$ of $R$ | | 11 |
| diagonalization | 8 | 47 |
| back transformation | 1 | |
| solve $\hat{x}\gamma = -z$ | 0 | 0 |
| Total | 58 | 107 |

Table 2
Comparison of CPU times (in msec.) for solving Example 2

| | PTLS | Classical TLS |
|---|---|---|
| bidiagonalization of $[A; B]$ | 15 | 15 |
| initialization $V$ | | 5 |
| diagonalization | 9 | 32 |
| back transformation | 2 | |
| compute $\Gamma$ and solve $\hat{X}\Gamma = -Z$ | 1 | 1 |
| Total | 27 | 53 |

0.3, 0.2 and 0.1. Hence, 3 solution vectors must be computed. $\epsilon = 10^{-13}$. 5 QR iterations were required for convergence. The CPU times obtained are those shown in Table 2.

## 4. Conclusions

Since the TLS solution of a set of equations $AX \approx B$ is deduced from only one right singular vector or in general, a basis of the right singular subspace associated with the smallest singular values of the data $[A; B]$, its computational cost can be considerably reduced by computing its SVD only partially.

Hereto, an improved algorithm PSVD was presented in a previous paper, computing the singular subspace of a matrix, associated with its smallest singular values.

Incorporating this 'partial SVD' algorithm into the TLS computations, results in an improved 'partial TLS' algorithm PTLS. Its higher efficiency w.r.t. the classical TLS algorithm is demonstrated and confirmed with computational results. Typically, PTLS reduces the computation time with a factor 2.

## References

[1] T.F. Chan, An improved algorithm for computing the singular value decomposition, *ACM Trans. Math. Software* **8** (1982) 72–83.
[2] J. Demmel and W. Kahan, Computing small singular values of bidiagonal matrices with guaranteed high relative accuracy, Tech. Report, Courant Institute, New York, October 13, 1987.
[3] J.J. Dongarra, J.R. Bunch, C.B. Moler and G.W. Stewart, *LINPACK User's Guide* (SIAM, Philadelphia, 1979).
[4] G.H. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, *SIAM J. Numer. Anal. Ser. B* **2** (1965) 205–224.
[5] G.H. Golub and C. Reinsch, Singular value decomposition and least squares solutions, in: J.H. Wilkinson and C. Reinsch, Ed., *Handbook for Automatic Computation, II, Linear Algebra* (Springer, New York, 1971) 403–420.
[6] G.H. Golub and C.F. Van Loan, An analysis of the total least squares problem, *SIAM J. Numer. Anal.* **17** (1980) 883–893.
[7] G.H. Golub and C.F. Van Loan, *Matrix Computations* (John Hopkins University Press, Baltimore, MD, 1983).
[8] C.L. Lawson and R.J. Hansen, *Solving Least Squares Problems* (Prentice Hall, Englewood Cliffs, NY, 1974).
[9] B.N. Parlett, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs, NJ, 1980).
[10] S. Van Huffel, Documented Fortran programs of the classical total least squares computation, the partial singular value decomposition and the partial total least squares computation, Int. report, ESAT-KUL 87/11, ESAT-Lab., Dept. Electr. Eng., K.U. Leuven, April 1987.

[11] S. Van Huffel, Analysis of the total least squares problem and its use in parameter estimation, Doctoral Dissertation, Dept. Electr. Eng., K.U. Leuven, June 1987.

[12] S. Van Huffel and J. Vandewalle, The use of total least squares techniques for identification and parameter estimation, Preprints 7th IFAC/IFORS Symposium on Identification and System Parameter Estimation, York, U.K., July 3–7, 1985, pp. 1167–1172.

[13] S. Van Huffel, J. Vandewalle and A. Haegemans, An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values, *J. Comput. Appl. Math.* **19** (1987) 313–330.